International Journal of Parallel, Emergent and Distributed Systems
Vol. 00, No. 00, January 2005, 1–18

Getting the most of WiFi Mesh Networks with 802.16 Mesh Emulation

Petar Djukic and Shahrokh Valaee[†‡]

(May 27, 2007)

Currently deployed wireless mesh networks are based on 802.11, WiFi, technology, which is not efficient in multihop scenarios. We present a method, which emulates 802.16 mesh networks over 802.11 hardware. The method works by embedding 802.16 packets into 802.11 broadcast packets and padding the 802.11 broadcast payload, so that the broadcasts are aligned to 802.16 TDMA frame boundaries. The method requires only software changes on the nodes using 802.11a for mesh communications. This means that the mesh networks installed with 802.11a hardware today can be upgraded with a software patch to take advantage of Quality-of-Service available in 802.16.

We use ns2 simulations to show the performance of the 802.11 based mesh networks with the embedded 802.16. We show that the hybrid system can achieve throughputs in multiples of what is possible with 802.11 hardware alone. First, the efficiency of the new system is significantly higher than the efficiency of 802.11 based systems, because we use broadcast packets. Second, the new system eliminates unnecessary collisions in the wireless channel since it takes advantage of scheduled wireless access with 802.16 mesh coordination function.

1 Introduction

Wireless mesh networks have proven to be a cost-effective way to interconnect access points spread out over a large geographical area. Wireless terminals connect to the access points on their first hop and their traffic is carried by the wireless mesh to the Point-of-Presence (POP) where it can go to the Internet. The POP is the only node in the network connected to the Internet and can also act as a base-station, or the mesh coordinator. The wireless backbone is made of mesh nodes designed to use inexpensive off-the-shelf parts such as IEEE 802.11 wireless cards, which use the free, license-exempt, radio spectrum.

Currently deployed mesh networks use 802.11 wireless devices for wireless

[†]The authors are with The Edward S. Rogers Sr. Department of Electrical and Computer Engineering University of Toronto, 10 King's College Road, Toronto, ON, M5S 3G4, Canada.
[‡]This work was sponsored in part by the LG Electronics Corporation.

Int. J. Parallel, Emergent and Distributed Systems ISSN 1744-5760 print/ ISSN 1744-5779 online
http://www.tandf.co.uk/journals
DOI: 10.1080/1744-5760xxxxxxxxxxxxx

mesh connectivity [1, 2]. However, 802.11 medium access control is not appropriate for commercial applications of mesh networks since the Distributed Coordination Function (DCF), used to coordinate 802.11 transmissions, cannot provide Quality of Service (QoS) [3]. IEEE is currently working on a new mesh standard, 802.11s, which defines a Mesh Coordination Function (MCF), designed specifically for multi-hop operation. The 802.11s MCF includes a Time Division Multiple Access (TDMA) mode, needed to provide guaranteed QoS in mesh networks [4]. Nevertheless, even when 802.11s standard is ratified and 802.11s hardware becomes available, it will be difficult and costly to upgrade the mesh networks installed today, since mesh networks are usually built with a large number of nodes.

We propose upgrading the software on the existing nodes with an implementation of 802.16 MCF that can work with the existing 802.11a hardware. IEEE 802.16 has already been ratified and it also uses TDMA with its own MCF [5]. Our method requires only software changes on the nodes using 802.11a for mesh communications. This means that the mesh networks installed with 802.11a hardware today can be upgraded with a software patch to take advantage of 802.16 MCF and do not have to wait for hardware upgrades to 802.11s. The importance of this approach is that it increases the operational lifetime of current mesh networks by providing them with TDMA QoS MAC layer.

Our approach has several advantages, rooted at the ability to re-use the 802.16 standard. First, there is no need to specify a new mesh overlay MAC protocol; the 802.16 standard gives a detailed description of the mesh protocol. The 802.16 MCF specifies the synchronization mechanism used to align all transmissions to frame boundaries, as well as the mechanism to negotiate TDMA allocations in each frame. In addition to the MCF, the 802.16 standard also specifies procedures for network entry and MAC layer encryption for mesh nodes, which are missing in 802.11s.

The software upgrade is implemented as an overlay MAC layer on the mesh nodes. We insert an 802.16 mesh driver between the network layer and the 802.11 driver. The 802.16 driver emulates the 802.16 MCF by packing packets coming from the network layer into 802.16 PDUs, which are then passed to the 802.11 network interface for transmission. Each 802.16 packet is embedded into an 802.11a broadcast packet, so that the resulting packets can be scheduled with the 802.16 MCF. In order to achieve true TDMA, necessary for the operation of 802.16 MCF, we force the 802.11 back-off procedure to use at most one slot by setting the 802.11 QoS parameter `CWmax=1`. Incoming 802.16 packets are also padded so that the resulting 802.11 packets can be aligned on TDMA boundaries.

We have implemented the 802.16 MCF and the 802.16 physical layer, as well as the 802.16 embedding scheme over 802.11 in the ns2 simulator [6]. To the best of our knowledge, this paper presents the first 802.16 mesh simulation

with the full TCP/IP protocol stack; the only other simulation with 802.16 that we are aware of is limited to the simulation of the 802.16 scheduling protocol [7]. Our simulations show that the embedded system can provide *guaranteed* link bandwidths despite the fact that it is running on top of 802.11 hardware.

We show that despite the embedding and the padding, our scheme is significantly more efficient than 802.11 DCF. The reason for the increased efficiency is that we eliminate RTS-CTS-ACK exchanges, used in 802.11 unicast transmissions, and replace them with 802.11 broadcasts, coordinated with 802.16 MCF. With 802.16 MCF, our embedding scheme also eliminates collisions from the wireless channel, adding further efficiency to 802.11 mesh networks. Our simulations show that with collisions eliminated, the throughput of the embedding scheme can be multiples of what is possible with 802.11 DCF alone. So, with a simple software upgrade the same 802.11 mesh network can support many times more users.

This work extends our prior work on the topic [8, 9] with deeper analysis of the embedding scheme and further simulations with ns2. The new simulations examine the performance of the embedded scheme with 802.16 scheduling algorithms proposed in literature [10, 11].

The rest of this paper is organized as follows. Section 2 describes 802.11 DCF and shows that 802.11 DCF has low efficiency due to the overhead of RTS-CTS-ACK exchanges. Section 3 describes 802.16 MCF and shows why it is appropriate for mesh networks. Section 4 shows how 802.16 MCF can be used with 802.11a hardware to achieve the performance of 802.16 MCF in 802.11 based mesh networks. Section 5 compares the performance of embedded 802.16 mesh networks with 802.11 mesh networks. Section 6 discusses practical issues in implementing the embedded scheme.

1.1 *Related Work*

In [12], the authors propose an overlay MAC layer for 802.11 mesh networks. The overlay uses admission control to ensure that real-time flows receive good quality of service, however the overlay itself does not improve the performance of the underlying 802.11 hardware. A different MAC overlay layer is proposed in [13]. This overlay improves the performance of 802.11 with loose TDMA synchronization. Even though the overlay MAC in [13] uses TDMA-like access to the wireless channel, it still only provides best effort service. In contrast, our approach uses small frame size and synchronization from the 802.16 protocol to provide *guaranteed* access to the channel.

The embedded 802.16 used in this paper can be used with software based MAC platforms like [14]. In that software platform, 802.11 operation is modified as far as the network driver will allow it, so that the network cards

behave almost as a software TDMA radio transceiver. On the other hand, our approach is appropriate even in the more general case, where the specific hardware used in [14] is not available, since our embedding does not make any assumptions about 802.11 hardware beyond what is specified in the standard [15].

2 802.11 Distributed Coordination Function

Current wireless mesh networks use IEEE 802.11a protocol to implement node-to-node connectivity [2, 15]. 802.11a achieves high raw bit rates by using an Orthogonal Frequency Division Multiplexing (OFDM) based physical layer in the 5GHz license-exempt band. However, even with the high bit rates, 802.11a is not the most appropriate protocol for mesh networks because of high overhead and the unfairness that can affect end-to-end flows. We first describe the 802.11 DCF and then show the efficiency of 802.11 DCF.

802.11 DCF uses collision avoidance to decrease the number of packet collisions in the network. Collision avoidance is based on the use of the basic procedure. In the basic procedure, the transmitter senses the channel until it becomes free. Once the channel is free, the transmitter waits for a duration of $T_{\text{DIFS}} = 34\mu$s [15] and then calculates an additional "back-off" waiting time. The back off time is chosen randomly in the interval of $[\texttt{CWmin}, \texttt{CWmax}] \times T_{\text{aSlotTime}}$ seconds (`CWmin` and `CWmax` are QoS parameters that can be specified by the user, while $T_{\text{aSlotTime}} = 9\mu$s [15]). After waiting for the end of the back-off time the transmitter sends a packet.

In addition to the basic procedure, the nodes also have the option of using RTS-CTS-ACK exchanges to coordinate their transmissions. The RTS and CTS are exchanged after the transmitter has performed the basic procedure. The exchange is used to announce to neighbouring nodes that a transmission will take place. As a part of the exchange, the transmitter announces how long the transmission will last, so that the neighbouring nodes can set their virtual sensing timers. After a successful packet reception, the receiver sends the ACK packet so that the sender knows the transmission was successful. Using the values for timing constants listed in the 802.11a standard [15], we find that the overhead of the full RTS-CTS-ACK exchanges is 318μs per packet.

The overhead of RTS-CTS-ACK exchanges significantly decreases the efficiency of 802.11a (Fig. 1). The efficiency is defined as the amount of time needed to transmit a packed on its own divided by the amount of time it takes to transmit the packet with the RTS-CTS-ACK exchange with no collisions (and no back-off). Later, we show that RTS-CTS-ACK exchanges are also not a good way to eliminate collisions in the wireless channel, which is the reason why a new coordination mechanism is necessary for multi-hop 802.11

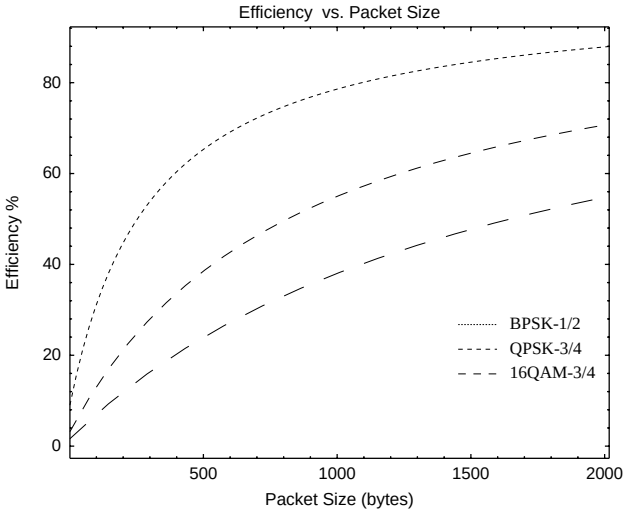


Figure 1. Efficiency of 802.11 DCF

mesh networks. Fig. 1 shows the efficiency as a function of packet size for BPSK-1/2,QPSK-3/4 and 16QAM-3/4 rates, corresponding to raw data rates of 6.0Mbps, 18.0Mbps, and 36.0Mbps, respectively. We see that the efficiency increases with the packet size and the efficiency decreases as the raw data rate increases. The low efficiency of RTS-CTS-ACK exchanges is the main reason that the 802.16 packet embedding described in Section 4 uses 802.11 broadcast packets, which do not use the RTS-CTS-ACK exchanges.

3 802.16 Mesh Coordination Function

IEEE 802.16 specifies a TDMA based MAC protocol [5] for mesh networks (802.16 MCF). In TDMA, the time is divided into slots of equal length and during each slot a block of bytes is broadcast. The slots are grouped into frames of equal length and the frames are then repeated over time. IEEE 802.16 MCF specifies how the slots in each frame are assigned to nodes.

First, we describe the physical layer used in 802.16 and compare it to the physical layer used in 802.11a. Both 802.16 and 802.11a use OFDM to achieve high data rates. However, 802.16 also uses TDMA to provide QoS. Second, we describe the 802.16 MCF and show its efficiency.

Table 1. Comparison of 802.11a and 802.16 Raw Data Rates

Modulation	Data $\frac{\text{Bits}}{\text{Symbol}}$		Bitrate $\frac{\text{Mbits}}{\text{second}}$		
	802.11a	802.16	802.11a 20MHz	802.16 10MHz	802.16 20MHz
BPSK-1/2	24	96	6.0	3.84	7.68
BPSK-3/4	36	X	9.0	X	X
QPSK-1/2	48	192	12.0	7.68	15.36
QPSK-3/4	72	288	18.0	11.52	23.04
16QAM-1/2	96	384	24.0	15.36	30.72
16QAM-3/4	144	576	36.0	23.04	46.08
64QAM-2/3	192	768	48.0	30.72	61.44
64QAM-3/4	216	864	54.0	34.56	69.12

3.1 *802.16 Physical Layer*

IEEE 802.16 uses OFDM in the licensed and license-exempt 5GHz frequency bands. OFDM transforms blocks of bits into constant duration symbols carried on a set of frequency orthogonal carriers. Since each OFDM symbol has the same duration, 802.16 uses them as TDMA slots. The raw data rate of 802.16 depends on the duration of each symbol (slot), which depends on the bandwidth, and the number of bits carried in each slot. The number of bits carried in a slot depends on the modulation scheme used during its transmission. Table 1 compares 802.16 with 10MHz and 20MHz bandwidths to 802.11a.[1]

3.2 *802.16 Frame Structure*

IEEE 802.16 organizes slots into frames for two reasons. First, the frame boundaries are used to synchronize the mesh nodes. Second, the frame structure allows the division of control and data traffic into sub-frames. In both data and control sub-frames, slots are grouped into *transmission opportunities*. The 802.16 MCF controls the transmission schedules in the control and data sub-frames. The control sub-frame is logically represented by a single channel, so a transmission schedule for the control sub-frame maps transmission opportunities to nodes. On the other hand, the data sub-frame has many logical channels each representing a link between two mesh nodes, so a transmission schedule in the data sub-frame maps transmission opportunities to links.

The size of the 802.16 control sub-frame is $7\times$`MSH-CTRL-LEN` slots, where each control sub-frame transmission opportunity is 7 slots long and

[1]IEEE 802.16 can use hardware with 10MHz, in the licensed 5GHz frequency band, or 20MHz, in the license-exempt 5GHz frequency band, while 802.11a uses 20MHz in the license-exempt 5GHz frequency band.

`MSH-CTRL-LEN` is a parameter left up to the network operator. Each of the `MSH-CTRL-LEN` transmission opportunities is occupied by a single control packet. Three slots are used to guard the packet, while the other four carry the payload. The packets in the control frame are sent at the lowest modulation, making the largest possible size for a control packet 48 bytes.

The data sub-frame is also divided into transmission opportunities with a fixed length. The length of data transmission opportunities is determined by dividing the number of OFDM symbols in the data sub-frame by 256.[1] The timing structure of data packets is similar to the structure of control packets, except that the data portion is variable. For example, if each transmission opportunity was 4 slots, the smallest data packet size at the lowest modulation, would be 12 bytes, where 3 slots are used for guard and only one slot is used for data. The next size for a data packet is 60 bytes for two transmission opportunities, and then 108 bytes for three transmission opportunities.

The control sub-frame transmission scheduling is specified by the standard. However, the standard leaves open how transmission schedules in the data sub-frame are determined and only specifies the mechanisms by which transmission schedules in the data sub-frame are announced. In the centralized scheduling protocol, nodes monitor the traffic demand from their subscribers and use this information to request bandwidth from the base-station. The base-station uses the requests to calculate the schedule for each link in the network. The schedule is then transmitted as a *tree* of links that the nodes should use to send packets to the base-station, together with the length of time each node should transmit in a frame. The schedule is flooded through the network, so that all mesh nodes know the entire transmission schedule. On the other hand, in the decentralized scheduling protocol nodes negotiate the starting transmission opportunity and duration for each link. This protocol uses handshakes to ensure all neighbouring nodes are aware of the transmission schedules.

Fig. 2 shows the efficiency of 802.16 operating at 20MHz. The efficiency is defined as the packet transmission time divided by the amount of time it takes to transfer the packet between two nodes (including MAC headers and guard times). We plot the efficiency as a function of packet size for BPSK-1/2, QPSK-3/4 and 16QAM-3/4 rates (Table 1). We see that the efficiency of 802.16 is much higher than the efficiency of 802.11a DCF (Fig. 1). As the packet size increases, the efficiency of all three rates converge to maximum much more quickly than in the case of 802.11a. The reason for this is that the overhead time is only 37.5μs (three 12.5μs slots) in the case of 802.16, whereas the overhead time for 802.11a is 318μs. Also the size of the 802.16 header is only 8 bytes compared to 22 bytes of the 802.11 header. IEEE 802.16

[1]The standard restricts the number of transmission opportunities in the data sub-frame to at most 256 because the duration fields in the scheduling control packets are 8 bits long.

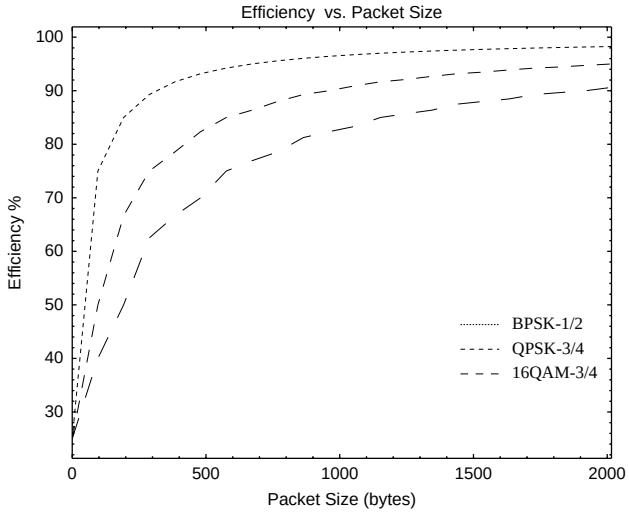


Figure 2. Efficiency of 802.16 MCF

has the smaller header because it does not transmit MAC addresses in each transmission. All that is needed to identify the transmitter and the receiver is the link identifier which is 2 bytes long.

4 802.16 Embedded over 802.11 Hardware

We proposed embedding 802.16 MAC packet data units (PDUs) into 802.11a MAC broadcast packets in [8, 9]. In that work, we have shown that the RTS-CTS-ACK exchanges are not necessary since 802.16 MCF coordinates the nodes, and acknowledgments are a part of the 802.16 protocol. We have shown in Section 2 that the RTS-CTS-ACK exchanges decrease the efficiency of the 802.11a protocol, so removing them actually makes our protocol more efficient than the 802.11a DCF.

Fig. 3 shows the embedding of the 802.16 packets into the 802.11a broadcast packets.[1] The embedding assumes that the back-off time is always one. For this to be true, we need to set `CWmin` = `CWmax` = 1.[2] The length of the PLCP preamble includes 12 training symbols as well as one OFDM symbol that carries the PLCP header, for a total of 13 symbols each with the duration

[1] It is possible to embed the packets in conjution with the sub-network access protocol (SNAP) [16]. However, this is not necessary under Linux because the kernel allows the definition of custom ethernet types, so we opted to use an embedding with a smaller size, i.e. without the SNAP sub-header.

[2] For real implementations, this assumptiondepends on the firmware support. For example, the Intel PRO/Wireless 2900bg cards allow this change [17].

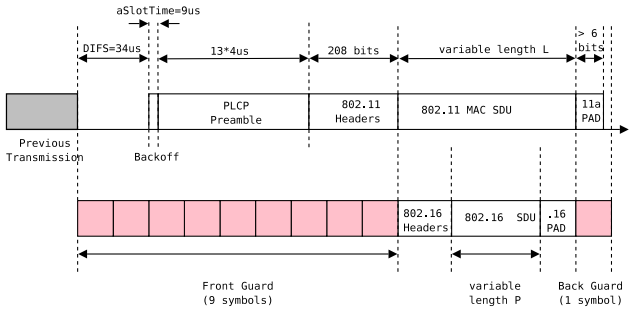


Figure 3. Embedding 802.16 PDU over 802.11a

of 4μs [18]. 802.11 headers include 2 bytes for the symbol field (specific to 802.11a), as well as 24 bytes for 802.11 MAC headers [15, 18].

The transmission time required to transfer L bits of the 802.16 embedded packet (Fig. 3) is given by :

$$T_{\text{tran}}^{802.11} = 95\mu\text{s} + \left\lceil \frac{214 + L}{r_{\text{mod}}^{802.11}} \right\rceil \times 4\mu\text{s}, \tag{1}$$

where 95μs is the total time required for the `DIFS` and `aSlotTime` times and the time required for 13 symbols of PLCP preamble, 214 bits of overhead comes from 802.11a headers (208 bits) and the minimum of 6-bit pad, and the $r_{\text{mod}}^{802.11}$ is the number of data bits in an 802.11 OFDM symbol.

The bottom part of Fig. 3 shows how 802.16 MAC interprets the embedded transmission. We set the slot duration $T_{\text{MS}} = 16\mu$s, so each slot corresponds to four 802.11a OFDM symbols. This slot duration maps one 802.16 slot to T_{MS}. We modify the settings for the 802.16 MAC protocol to make the front guard time 9 slots and the back guard time one slot. We choose the value of 9 for the front guard because that is longer than the time required to transmit all of 802.11a overhead with any modulation rate. The number of back guard slots is the minimum required to transmit 802.11a padding.

The 802.16 overlay MAC layer transmission time is:

$$T_{\text{tran}}^{802.16} = \left[\left\lceil \frac{P + 64}{r_{\text{mod}}^{802.16}} \right\rceil + 10 \right] \times 16\mu\text{s} > T_{\text{tran}}^{802.11}, \tag{2}$$

where $r_{\text{mod}}^{802.16}$ is the number of data bits in an 802.16 symbol, P is the size of the network layer packet in bits and 64 bits of overhead come from the 802.16 MAC headers. Because of the number of guard slots, the transmission time seen by the 802.16 overlay MAC is strictly greater than the time required to transmit the packet over 802.11a, ensuring 802.11a collisions do not occur.

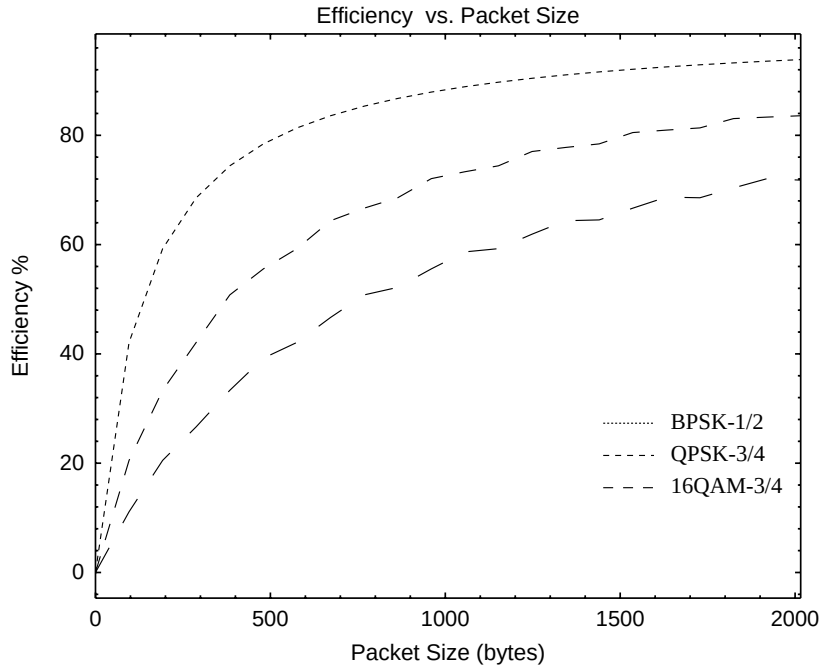


Figure 4. Efficiency of embedded 802.16 MCF

If a corrupted packet is received by a node, the DCF on the receiver node selects a timeout of $T_{\text{EIFS}} = 658\mu$s before the basic procedure can start on that node. This timer is cancelled if another packet is received correctly before the timer expires. The embedding module always checks if the previous transmission was successful before transmitting a new packet, and it cancels all packets requested by the 802.16 module during active transmissions. This procedure makes sure that if an EIFS timer is on at a transmitter node, the packets from that node do not get delayed past the start time of the next transmitting node. At most one packet from any transmitter will colide with packets from another node. If a collision happens, the DCF back-off mechanism cannot increase `CWmax` more than 1 so (2) will still hold.

Fig. 4 shows the efficiency of the embedded 802.16. We plot the efficiency as a function of packet size for BPSK-1/2, QPSK-3/4 and 16QAM-3/4 rates (Table 1). We see that the efficiency of 802.16 is higher than the efficiency of 802.11a DCF (Fig. 1), however it is lower than the efficiency of 802.16 (Fig. 2). This supplements our observations from the previous section. The efficiency of the embedded scheme is smaller than 802.16 because the overhead of transmitting the packet is higher, e.g. 10 symbols compared to three symbols for 802.16. The efficiency of the embedded scheme is smaller than 802.11 because

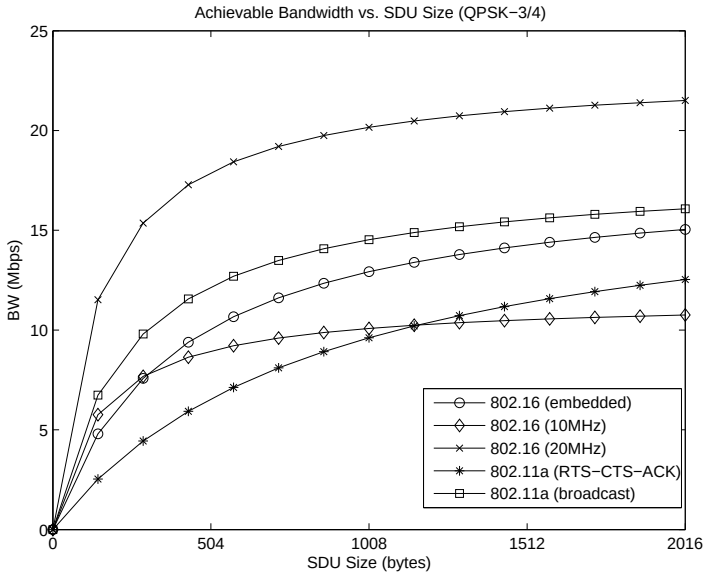


Figure 5. Comparison of Data Rates

the duration of the 10 extra symbols is shorter than the 318μs added in each transmission using RTS-CTS-ACK exchanges.

4.1 *Bandwidth of the embedded 802.16*

Fig. 5 compares the bandwidth of the embedded transmissions to the bandwidths that can be achieved using 802.16 and 802.11a when the modulation is QPSK-3/4. The bandwidth is defined as the number of bits divided by the actual time required to transmit those bits. The bandwidth for 802.11a DCF is found by assuming that there are no collisions in the channel and the sender and receiver use RTS-CTS-ACK exchanges. This is a very optimistic view of the 802.11 bandwidth.

First, we observe that the achievable bandwidth of the embedded scheme ("802.16 (embedded)") is comparable to the bandwidth that can be achieved using 802.16 hardware. The achievable bandwidth of the embedded scheme is always less than what can be achieved with 802.16 hardware operating at 20MHz ("802.16 (20Mhz)"), and more than the bandwidth of 802.16 hardware operating at 10MHz ("802.16 (10Mhz)"). This corresponds to the fact that the actual symbol duration is 16μs in the embedded scheme compared to 12.5μs and 25μs for 802.16 hardware operating at 20MHz and 10MHz, respectively.

Second, we observe that the achievable bandwidth is higher than the achiev-

able bandwidth of 802.11a DCF ("802.11 (RTS-CTS-ACK)"). This is because the embedded scheme has less overhead even with the added padding. In fact, the 802.11a bandwidth shown in the figure is a very optimistic view of 802.11a bandwidth since it does not take into account collisions, nor regular back-offs. In practice, the throughput of 802.11a is much lower, especially with multiple active connections.

Third, the achievable bandwidth of the embedded 802.16 is lower than the bandwidth that could be achieved by using broadcasts only ("802.11 (broadcast)"). This is due to the padding of 802.11a broadcast packets. Again, the bandwidth shown for the 802.11a broadcast is optimistic since we are not accounting for back-off time.

5 Simulation Results

In this section we examine the performance of the MAC layers discussed in the previous sections with ns2 [6] simulations. We have changed the ns2 simulator to include 802.11a physical layer, 802.16 with its physical layer and 802.16 embedded over 802.11a. For the simulations using 802.16 MCF, we have also implemented 802.16 scheduling algorithms proposed in [10, 11].

In all of our scenarios, the traffic is setup the way it would be in a typical mesh network with the mesh nodes connecting to the base-station (point-of-presence). Each node sets up an FTP connection to the base-station and the base-station sets up an FTP connection to each each mesh node. Each FTP connection is used to transfer a large file (10Mb). Since the network is static it does not make sense to use any ad-hoc network routing protocols such as AODV [19] or DSR [20], rather the routes are chosen with a simple minimum-hop distance criterion. This is in-line with the current implementations of mesh networks, which use OSPF [2, 4].

In our simulations, we use the 802.16 physical layer operating at 20MHz since the standard mandates this for the license-exempt 5GHz band, which is also used by 802.11a. We use 20ms frame size. With this frame duration, 802.16 operating at 20MHz has 1600, 12.5μs, long slots in each frame, while 802.16 embedded in 802.11a has has 1250, 16μs slots. The first, 35 slots of the frame are allocated for the control sub-frame (5 control sub-frame transmission opportunities). The rest of the slots are allocated for the data sub-frame, in terms of 7 slot long transmission opportunities for 802.16 operating at 20MHz and 5 slot long transmission opportunities for 802.16 embedded over 802.11a hardware. For simplicity, we use centralized scheduling only, so the entire data sub-frame is allocated with the centralized scheduling protocol.

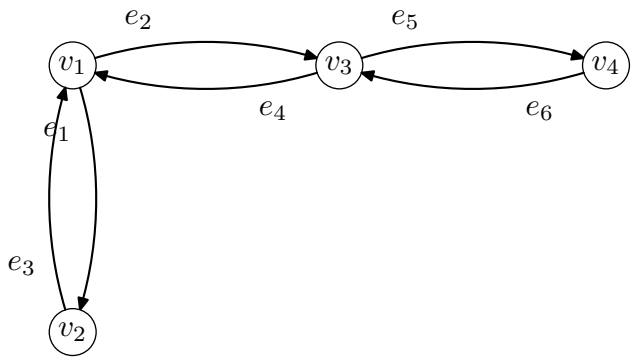
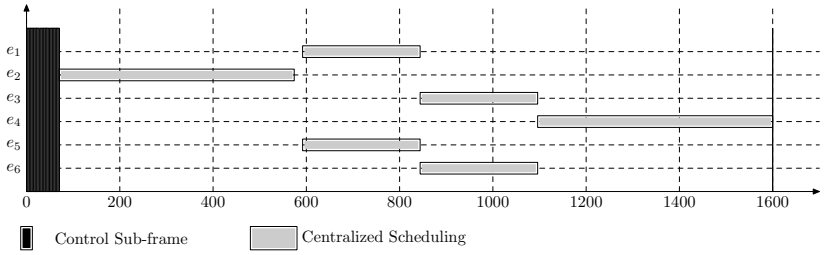


Figure 6. Example Mesh Network

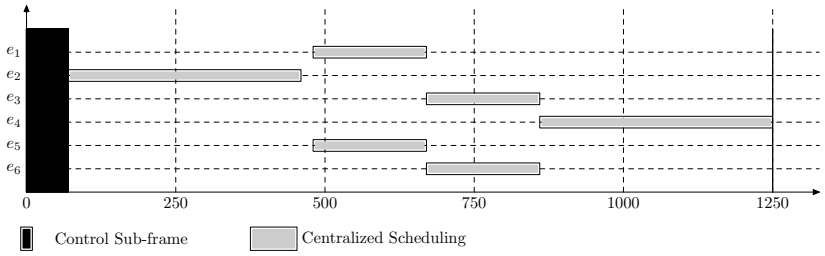
5.1 *Example 1: Simple Topology*

Fig. 10 shows a small mesh network we use to examine the performance of our embedding scheme. Node v_1 acts as the base-station. All links use the modulation of QPSK-3/4. We use the Bellman-Ford TDMA scheduler [11] to find the TDMA schedule for links in the network (Fig. 7). Full description of the the scheduler when applied to 802.16 is available in our survey of 802.16 scheduling algorithms [10]. The scheduler finds the TDMA schedule that maximizes the end-to-end bandwidth, while also minimizing end-to-end TDMA delay of packets traversing the network. TDMA scheduling delay occurs when packets arriving on an inbound link must wait for the subsequent frame to be transmitted on the outbound link [10]. In the case of the network in Fig. 10, the scheduler is able to allocate uplink and downlink bandwidths 384kbytes/second for each node in the case of 802.16 embedded over 802.11a hardware and 484kbytes/second for 802.16 operating at 20MHz. In both cases, the round trip TDMA scheduling delay is 20ms for each node.

We compare the total accumulated traffic of 802.16 operating at 20MHz, our embedding scheme and 802.11a for the first 100 seconds of the simulation in Fig. 8. We also compare total throughput of each node (uplink and downlink) during the first 100 seconds of the simulation in Fig. 9. We use the same naming scheme as in Fig. 5. We see that 802.16 performs better than our embedding scheme. This is because our embedding technology is limited by the overhead of 802.11a. All simulations use maximum transmission unit size of 512 bytes, which results in 802.16 operating at 20Mhz to be about 70% more efficient than 802.16 embedded over 802.11a (Fig. 5). We see from Fig. 9 that throughput of 802.16 802.16 operating at 20Mhz is 68% higher than the throughput of 802.16 embedded over 802.11a consistent with our observations in Fig. 5. However, our embedding scheme still performs significantly better than 802.11a, because it does not waste bandwidth on unnecessary collisions.



(a) 802.16 @ 20MHz



(b) 802.16 Embedded Over 802.11a

Figure 7. 802.16 Schedule used for the topology in Fig. 10

In addition to increasing total throughput by 33%, our scheme also ensures that the wireless channel is used more fairly than in 802.11a. For example, with our embedding scheme throughput of node 4 improves more than 3 times (Fig. 9).

5.2 *Example 2: Chains*

In this section, we examine performance of scheduling algorithms on chain topologies. In each scenario, we create a chain of mesh nodes where one of the end nodes is the base-station and all other nodes in the topology are regular mesh nodes. We use the Bellman-Ford scheduler to find all transmission schedules for 802.16 MCF simulations. We plot total throughput, achieved by all nodes, in Fig. 10. We see that our embedding scheme performs at least almost 3 times better than 802.11a for the 3 node chain and almost 5 times better than 802.11a for the 6 node chain. The performance gain is significantly higher than what is shown in Fig. 5 because 802.16 MCF eliminates collisions that would normally exist if 802.11a hardware was used on its own.

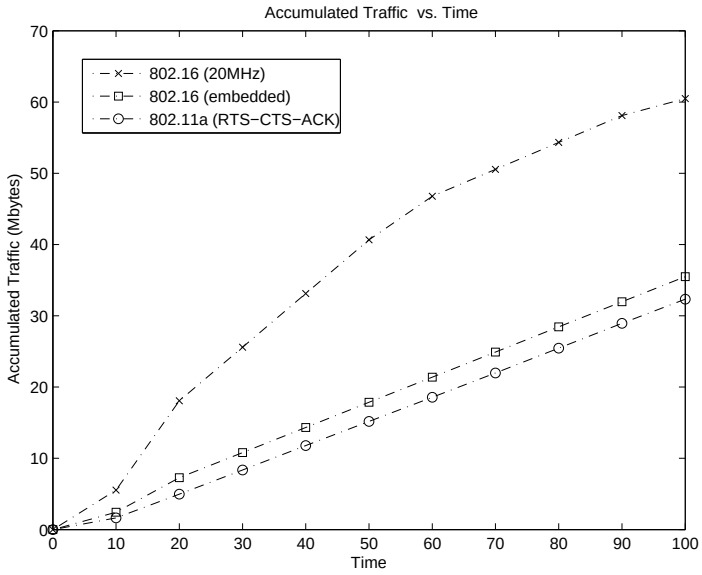


Figure 8. Cumulated traffic

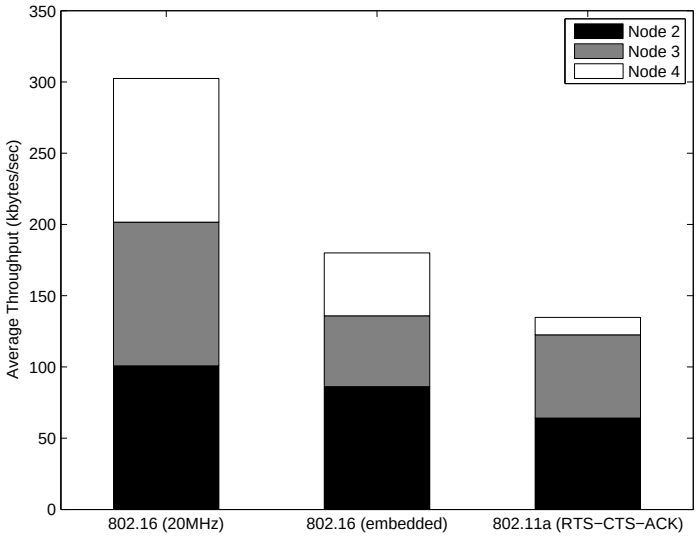


Figure 9. Throughput

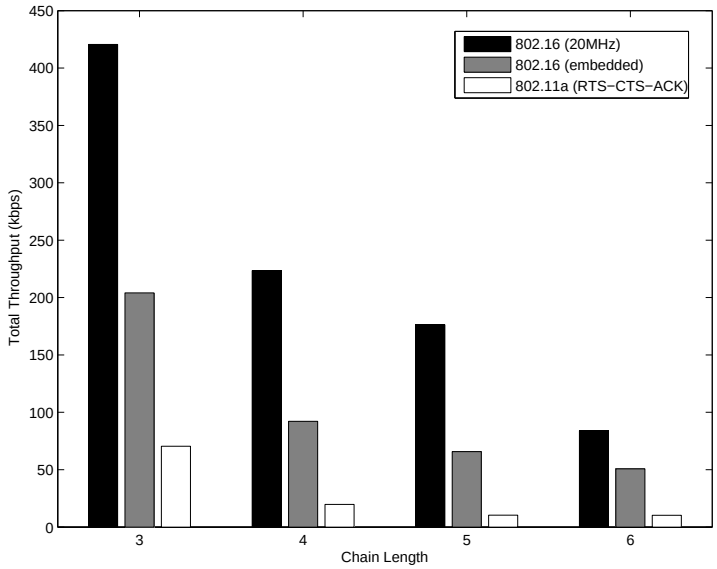


Figure 10. Throughput for Chain Topology

6 Implementation under Linux OS

In this section, we show how we implement the embedded 802.16 transmission described in Section 4. The 802.16 module used in the ns2 simulations was made in a way that makes it easily portable to Linux. We are currently integrating the module to the Linux kernel [21].

Fig. 11 shows the architecture of the Linux kernel implementation. We are adding a Linux kernel module, that behaves like a network card ("802.16 Driver"). The module binds itself to a real wireless card, and acts as a proxy on behalf of that card. The module is also linked in with the 802.16 module ("802.16 implementation"), which we have already tested with ns2. Packets coming from the network layer are first accepted by the 802.16 driver, and then passed to the 802.16 implementation. The 802.16 implementation queues up the packets until they should be transmitted according to the schedule for connection that the packets are traversing. When packets come from the wireless card, Linux kernel transfers them to the 802.16 module where they are filtered out, stripped of 802.16 header data, and passed on to the network layer. The kernel module also interacts with the OSPF routing program on the node. Since the 802.16 protocol is connection oriented, the OSPF should be notified when a new node joins the network, and the connection to that node is created. The kernel module sends a message to OSPF to indicate the presence of a new link to a neighbour.

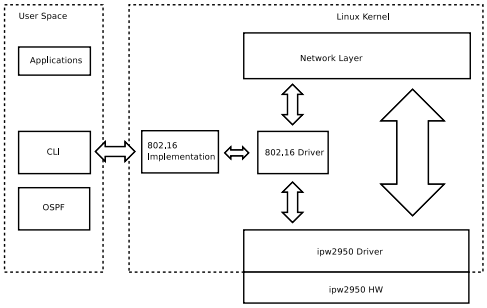


Figure 11. Linux Implementation

In addition to the kernel module, we are develop a CLI interface program that resides in user space and that forwards management Service Access Point primitives to the 802.16 implementation. The CLI interface supports the commands necessary to manage the 802.16 module. This interface is very similar to the scripting commands used in ns2.

The Linux implementation has presented us with several challenges not present when we integrated our 802.16 module into ns2. First, the Linux kernel does not provide memory allocation functions available in user space. This meant that the whole 802.16 module had to be ported into C from C++, with its own memory management, and re-tested again in ns2. Second, the Linux kernel default timer support is in the order of milliseconds and we need microsecond support for our 802.16 module. This means that we have to use Linux distributions with real-time support and microsecond clock precission, and divert from the mainstream Linux kernel distribution. Third, we have the requirement that the wireless card should support changing of `CWmin` and `CWmax`. We have found that the linux driver for the Intel PRO/Wireless 2200BG card has a QoS interface, which allows setting of `CWmin` and `CWmax` [17].

7 Conclusion

We have presented a method that allows 802.16 MCF to be used for TDMA data transmissions in 802.11a mesh networks. The method works by embedding 802.16 packets into 802.11a broadcast packets with padding so that all transmissions can be aligned on 802.16 frame boundaries. Our method significantly decreases the overhead inherent in RTS-CTS-ACK exchanges, which are a part of the 802.11 DCF. At the same time, our method removes collisions from the channel because the nodes are communicating in 802.16 TDMA fashion. We have used ns2 simulations to show that if our method is used in 802.11a based mesh networks, the resulting hybrid MAC can achieve throughput in

multiples of what is possible with 802.11a hardware alone.

Our method requires only software changes on the nodes using 802.11a for mesh communications. This means that the mesh networks installed with 802.11a hardware today can be upgraded with a software patch to take advantage of 802.16 MCF and do not have to wait for hardware upgrades to 802.11s.

References

[1] J. Camp, J. Robinson, C. Steger, and E. Knightly, "Measurement driven deployment of a two-tier urban mesh access network," Rice University, Technical Report TREE0505, December 2005.

[2] Nortel Networks, "Wireless mesh network - extending the reach of wireless LAN, securely and cost-effectively," http://www.nortelnetworks.com/solutions/wlan/, November, 2003.

[3] S. Xu and T. Saadawi, "Does the IEEE 802.11 MAC protocol work well in multihop wireless ad hoc networks," *IEEE Communications Magazine*, vol. 39, no. 6, pp. 130–137, June 2001.

[4] "802.11 TGs MAC enhacement proposal," IEEE Protocol Proposal I802.11-05/0575r3, September 2005.

[5] "IEEE standard for local and metropolitan area networks part 16: Air interface for fixed broadband wireless access systems," 2004.

[6] "http://www.isi.edu/nsnam/ns/."

[7] M. Cao, W. Ma, Q. Zhang, X. Wang, and W. Zhu, "Modeling and performance analysis of the distributed scheduler in IEEE 802.16 mesh mode," in *MobiHoc*, 2005, pp. 78–89.

[8] P. Djukic and S. Valaee, "802.16 MCF for 802.11a based mesh networks: A case for standards re-use," in *Proceedings of 23rd Queen's Biennial Symposium on Communications*, 2006.

[9] ——, "Towards guaranteed QoS in mesh networks: Emulating WiMAX mesh over WiFi hardware," in *The Fourth Workshop on Wireless Ad hoc and Sensor Networks WWASN*, 2007.

[10] ——, "Scheduling algorithms for 802.16 mesh networks," in *Wimax/MobileFi: Advanced Research and Technology*, Y. Xiao, Ed. Auerbach Publications, CRC Press, 2007.

[11] ——, "Link scheduling for minimum delay in spatial re-use TDMA," in *Proceedings of INFOCOM*, 2007.

[12] H. Wu, Y. Liu, Q. Zhang, and Z.-L. Zhang, "SoftMAC: Layer 2.5 collaborative MAC for multimedia support in multi-hop wireless networks," *IEEE Transactions on Mobile Computing*, to Appear.

[13] A. Rao and I. Stoica, "An overlay MAC layer for 802.11 networks," in *MobiSys*, 2005, pp. 135–148.

[14] M. Neufeld, J. Fifield, C. Doerr, A. Sheth, and D. Grunwald, "SoftMAC—flexible wireless research platform," in *HotNets*, 2005.

[15] "IEEE standard for local and metropolitan area networks part 11: Wireless lan medium access control (MAC) and physical layer (PHY) specifications," 1999.

[16] "IEEE standard for local and metropolitan area networks: Overview and architecture," 2001.

[17] "Intel® PRO/Wireless 2200BG driver for linux," http://ipw2200.sourceforge.net/.

[18] "IEEE standard for local and metropolitan area networks part 11: Wireless LAN medium access control (MAC) and physical layer (PHY) specifications high-speed physical layer in the 5 GHz band," 1999.

[19] C. E. Perkins and E. M. Royer, "Ad hoc on-demand distance vector routing," in *Proceedings of the 2nd IEEE Workshop on Mobile Computing Systems and Applications*, New Orleans, LA, February 1999, pp. 90–100.

[20] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*, ser. The Kluwer International Series in Engineering and Computer Science, Imielinski and Korth, Eds. Kluwer Academic Publishers, 1996, vol. 353.

[21] "Linux kernel," http://www.kernel.org/, 2006.